TEMA 4
El tipo conjunto

PROGRAMACIÓN Y ESTRUCTURAS DE DATOS

Tipo conjunto

1. Tipo Conjunto
DEFINICIONES

- **Un conjunto es una colección de elementos, cada uno de los cuales puede ser un conjunto, o un elemento primitivo que recibe el nombre de átomo**
- **Todos los miembros del conjunto son distintos**
- **El orden de los elementos no es importante (distinto de las listas)**

Notación de Conjuntos

- Se representa encerrando sus miembros entre llaves {1,2,5}
- Relación fundamental, la de pertenencia: $\in$ {x / x $\in$ Naturales} , {x / x < 8}
- Existe un conjunto especial sin elementos: $\varnothing$
- A $\subset$ B si todo elemento de A también lo es de B
- Conjunto Universal: formado por todos los posibles elementos que puede contener

1. Tipo Conjunto
SINTAXIS

MODULO GENERICO ModuloConjunto
MODULO Conjunto **USA** Boolean, Natural
SINTAXIS

Crear ( ) → Conjunto
Insertar(Conjunto, Ítem) → Conjunto
Eliminar(Conjunto, Ítem) → Conjunto
Pertenece(Conjunto, Ítem) → Boolean
EsVacíoConjunto(Conjunto) → Boolean
Cardinalidad(Conjunto) → Natural
Unión(Conjunto, Conjunto) → Conjunto
Intersección(Conjunto, Conjunto) → Conjunto
Diferencia(Conjunto, Conjunto) → Conjunto

VAR

C, D: Conjunto; x, y: Ítem;

1. Tipo Conjunto
SEMÁNTICA (I)

```
EsVacíoConjunto( Crear ) ←→  Cierto
EsVacíoConjunto( Insertar( C, x ) ) ←→  Falso
Insertar( Insertar( C, x ), y ) ←→
        si ( x == y ) entonces Insertar( C, x )  //no se permiten elementos repetidos
        sino Insertar( Insertar( C, y ), x )     //da igual el orden de inserción de los elem.
Eliminar( Crear, x ) ←→ Crear
Eliminar( Insertar( C, x ), y ) ←→
        si ( x == y ) entonces C                 // ¿y si permitieran elementos repetidos?
        sino Insertar( Eliminar( C, y ), x )
Pertenece( Crear, x ) ←→ Falso
Pertenece( Insertar( C, x ), y ) ←→
        si ( x == y ) entonces Cierto
        sino Pertenece( C, y )
Cardinalidad( Crear ) ←→  0
Cardinalidad(Insertar(C,x)) ←→  1+Cardinalidad(C)
Unión( Crear, C ) ←→  C
Unión( Insertar( C,x ), D ) ←→
        si ( Pertenece( D, x ) ) entonces Unión( C, D )
        sino Insertar( Unión( C, D ), x )
```

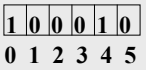

1. Tipo Conjunto
SEMÁNTICA (II)

```
Diferencia( Crear, C ) ←→ Crear
Diferencia( Insertar( C,x ), D ) ←→
        si ( Pertenece( D, x ) )
        entonces Diferencia( C,  D )
        sino Insertar( Diferencia( C, D ), x )
Intersección( Crear, D ) ←→ Crear
Intersección( Insertar( C,x ), D ) ←→
        si ( Pertenece( D, x ) )
        entonces Insertar( Intersección( C, D ), x )
        sino Intersección( C,  D )
```

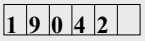
1. Tipo Conjunto
IMPLEMENTACIÓN

- **Mediante un vector**

  -Vector de bits/enteros (cada componente corresponde a un elemento del conjunto universal)

  -Vector de elementos

- **Almacenar los elementos conforme se inserten (mediante listas, árboles, ...):**

  Espacio proporcional al conjunto representado

1. Tipo Conjunto
EJERCICIO

- **Rellenar la siguiente tabla de complejidades (peor caso):**

m=elem. conjto. n=elem. conjto. Univ.	Vector de Bits	Lista ordenada	Lista desordenada
Búsqueda			
Inserción			
Unión			

2. DICCIONARIO
DEFINICIÓN

- **Subtipo del CONJUNTO, con las operaciones:**
  - **CREAR**
  - **INSERTAR**
  - **BORRAR**
  - **BÚSQUEDA**

2. DICCIONARIO
IMPLEMENTACIÓN

Implementaciones sencillas:

- **Mediante listas o vectores**
  Búsqueda, Inserción y Borrado:
  Listas: O (n)
  Vector Bits: O (1)
  Vector Elementos: O (n)

- **Mediante TAD Tabla de Dispersión (HASHING)**

2.1. TABLA DE DISPERSIÓN (HASHING)

DEFINICIÓN

- **HASHING: Utilizaremos la información del elemento a almacenar para buscar su posición dentro de la estructura**

- **Operaciones:**
  - **Búsqueda. O(1)**
  - **Inserción. O(1)**
  - **Borrado. O(1)**

2.1. TABLA DE DISPERSIÓN (HASHING)

MÉTODO

- Dividir el conjunto en un número finito "B" de clases
- Se usa función de dispersión H, tal que H(x) será un valor entre 0 y B-1

Formas de dispersión:

Abierta: No impone tamaño límite al conjunto

Cerrada: usa un tamaño fijo de almacenamiento (limita el tamaño)

2.1. Tabla Hash. Dispersión Cerrada

DEFINICIÓN

- **Los elementos se almacenan en tabla de tamaño fijo (TABLA DE DISPERSIÓN)**
- **La tabla se divide en B clases, y cada una podrá almacenar S elementos**
- **La Función de dispersión se implementa mediante una función aritmética**

$$H(x) = x \ MOD \ B$$

2.1. Tabla Hash. Dispersión Cerrada

INSERCIÓN

Caso COLISIÓN: x1, x2 (SINÓNIMOS/ H(x1) = H(x2))

ESTRATEGIA DE REDISPERSION:

• Elegir sucesión de localidades alternas dentro de la tabla, hasta encontrar una vacía

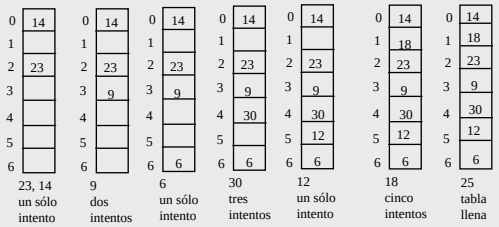
H(x), h1(x), h2(x), h3(x), ...

• Si ninguna está vacía: no es posible insertar

2.1. Tabla Hash. Dispersión Cerrada

INSERCIÓN. EJEMPLO

Ejemplo. Insertar en una tabla de dispersión cerrada de tamaño *B=7*, con función de dispersión *H(x)= x MOD B*, y con estrategia de redispersión *la siguiente posición de la tabla*, los siguientes elementos: *23, 14, 9, 6, 30, 12, 18, 25*



Nº TOTAL DE INTENTOS HASTA LA CLAVE 18:

14

2.1. Tabla Hash. Dispersión Cerrada

BÚSQUEDA. BORRADO

BÚSQUEDA DE ELEMENTOS

Buscar en sucesión de localidades alternas dentro de la tabla, hasta encontrar una vacía:

H(x), h1(x), h2(x), h3(x), ...

BORRADO DE ELEMENTOS

Hay que distinguir durante la búsqueda:

- Casillas vacías
- Casillas suprimidas

Durante la inserción las casillas suprimidas se tratarán como espacio disponible.

2.1. Tabla Hash. Dispersión Cerrada

ANÁLISIS (I)

- *ESTRATEGIA DE REDISPERSIÓN LINEAL ("siguiente posición")*:

  - No eficiente. Larga secuencia de intentos

  $h_i(x) = (H(x) + 1 \bullet i) \text{ MOD } B$ / c=1 $\quad h_i(x) = (h_{i-1}(x) + 1) \text{ MOD } B$

- *ESTRATEGIA DE REDISPERSIÓN ALEATORIA*:

  $h_i(x) = (H(x) + c \bullet i) \text{ MOD } B$ / c>1 $\quad h_i(x) = (h_{i-1}(x) + c) \text{ MOD } B$

  Sigue produciendo AMONTONAMIENTO: siguiente intento sólo en función del anterior

  c y B no deben tener factores primos comunes mayores que 1

- *E.R. CON 2ª FUNCION DE HASH*:

  $k(x) = (x \text{ MOD } (B-1)) + 1$

  $h_i(x) = (H(x) + k(x) \bullet i) \text{ MOD } B$ $\quad h_i(x) = (h_{i-1}(x) + k(x)) \text{ MOD } B$

  B debe ser primo

2.1. Tabla Hash. Dispersión Cerrada

ANÁLISIS (II)

- *LA MEJOR FUNCIÓN DE DISPERSIÓN*:
  - Que sea fácil de calcular
  - Que minimice el nº de colisiones
  - Que distribuya los elementos de forma azarosa
  - Debe hacer uso de toda la información asociada a las etiquetas

2.1. Tabla Hash. Dispersión Cerrada

ANÁLISIS (III)

- **Estrategia de redispersión aleatoria**
  - c y B no deben tener factores primos comunes mayores que 1 para que busque en todas las posiciones de la tabla
    - Ejemplo→ c=4; B=6

      $h_i(x) = (H(x) + c \cdot i) \text{ MOD } B = (h_{i-1}(x) + c) \text{ MOD } B$

      $H(10)=10 \text{ MOD } 6=4$

      $h_1(10)=(4+4 \cdot 1) \text{ MOD } 6=(4+4) \text{ MOD } 6=2$

      $h_2(10)=(4+4 \cdot 2) \text{ MOD } 6=(2+4) \text{ MOD } 6=0$

      $h_3(10)=(0+4) \text{ MOD } 6=4$; $h_4(10)=(4+4) \text{ MOD } 6=2$

X		X		X	
0	1	2	3	4	5

    - Ejemplo→ ¿c=6; B=9? ¿c=2; B=9?
- **Estrategia de redispersión con 2ª función hash**
  - B debe ser primo para que busque en todas las posiciones de la tabla
    - c=k(x) → 1…B-1 // k (x) = (x MOD (B-1)) + 1

      $h_i(x) = (H(x) + k(x) \cdot i) \text{ MOD } B = (h_{i-1}(x) + k(x)) \text{ MOD } B$

      B=7; k(x)=1…6

      B=11;k(x)=1…10

2.1. Tabla Hash. Dispersión Cerrada

EJERCICIOS

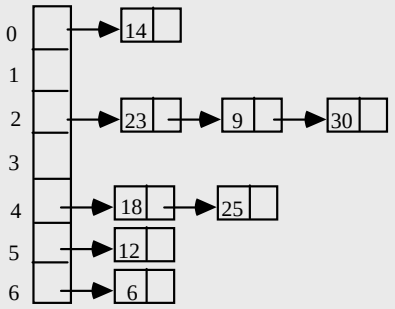
1) Insertar en una tabla de dispersión cerrada de tamaño *B=7*, con función de dispersión *H(x)= x MOD B*, y con estrategia de redispersión *segunda función hash*, los siguientes elementos: *23, 14, 9, 6, 30, 12, 18*

2.1. Tabla Hash. Dispersión Abierta

DEFINICIÓN

- Elimina el problema del CLUSTERING SECUNDARIO (colisiones entre claves no sinónimas)

- Las colisiones se resuelven utilizando una lista enlazada



2.1. Tabla Hash

FACTOR DE CARGA (I)

$$\alpha = \frac{n}{|B|}$$

n = nº elem. de la tabla. B = tamaño de la tabla

HASH CERRADO: $0 \leq \alpha \leq 1$
HASH ABIERTO: $\alpha \geq 0$ (No hay límite en el nº de elementos en cada casilla).

Teorema:

En **Hash Abierto** con factor de carga α:

- el nº esperado de pruebas en inserción o búsqueda sin éxito es '$1+\alpha$', y

- el nº esperado de pruebas para borrado o búsqueda con éxito es '$1+1/2 \cdot (1 + \alpha)$'

2.1. Tabla Hash

FACTOR DE CARGA (II)

Teorema:

En **Hash Cerrado con resolución lineal de colisiones**, con factor de carga α:

• el nº esperado de pruebas en inserción o búsqueda

sin éxito es $\frac{1}{2} \bullet \left(1 + \left(\frac{1}{1 - \alpha}\right)^2\right)$, y

• el nº esperado de pruebas para borrado o búsqueda

con éxito es $\frac{1}{2} \bullet \left(1 + \frac{1}{1 - \alpha}\right)$

2.1. Tabla Hash

FACTOR DE CARGA (III)

Teorema:

En **Hash Cerrado con resolución aleatoria de colisiones**, con factor de carga α:

• el nº esperado de pruebas en inserción o búsqueda

sin éxito es $\frac{1}{1 - \alpha}$, y

• el nº esperado de pruebas para borrado o búsqueda

con éxito es $\frac{-1}{\alpha} \bullet \log(1 - \alpha)$

2.1. Tabla Hash

FACTOR DE CARGA (IV)

E: Nº Esperado de Intentos.
c.éx: con éxito. **s.éx**: sin éxito.

	H.C.L.		H.C.Aleat.		H.Abierto	
α	**E c.éx**	**E s.éx.**	**E c.éx**	**E s.éx.**	**E c.éx**	**E s.éx.**
0.1	1.06					
0.25	1.17					
0.5	1.50					
0.75	2.50	8.5	1.9	4.0	1.8	2.0
0.9	5.50	50.5	2.6	10.0	1.9	2.0
0.95	10.50					

2.1. Tabla Hash

COMPARACIÓN HASH ABIERTO Y CERRADO

- H.A. es más eficiente y con menor degradación (cuanto más lleno funciona mejor que el H.C.)

- H.A. requiere espacio para los elementos de la lista, por lo que H.C. es más eficiente espacialmente

- Reestructuración de las tablas de dispersión:
  $n \geq 0{,}9\ B$ (H.C.)
  $n \geq 2\ B$ (H.A.)

→ Nueva tabla con el doble de posiciones